Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Decision Support Systems 40 (2005) 269–281
Decision Support Systems
www.elsevier.com/locate/dsw

Decision Support System induced guidance for model formulation and solution

Reza Barkhi[a,*], Erik Rolland[b], John Butler[c], Weiguo Fan[a]

[a] *Department of Accounting and Information Systems, Pamplin College of Business, Virginia Polytechnic Institute and State University, Pamplin Hall, Blacksburg, VA 24061, USA*
[b] *The A. Gary Anderson Graduate School of Management, University of California, Riverside, CA 92521, USA*
[c] *Department of Accounting and MIS, Fisher College of Business, The Ohio State University, Columbus, OH 43210, USA*

Received 1 January 2003; accepted 1 January 2004
Available online 26 April 2004

Abstract

One of the critical functions of Decision Support System (DSS) is to provide system induced decision guidance for proper model formulation and solution. We show how to incorporate this type of system induced decision guidance into the design of the next generation of DSS. We suggest that a DSS should make decisions, or at least recommendations, regarding what models should be executed to solve problems most effectively and this information should be generated inductively and used deductively. This information then becomes the meta-model to induce the user to make appropriate choices. We provide an example that will illustrate how two specific problem characteristics, namely the tightness of constraints and the linearity of constraints, influence the solution quality and solution times for a specific class of test problems. We argue that a DSS should execute different formulations of the problem that lead to satisficing solutions guiding DSS users in finding the best approach to solve complex problems.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Decision Support Systems; Decision modeling

1. Introduction

Despite the major developments in intelligent search techniques such as Genetic Algorithms (GA) and Simulated Annealing (SA), few Decision Support Systems (DSS) have incorporated these techniques as an integral part of their design. Both GA and SA are meta-heuristic search techniques and can be viewed as knowledge discovery techniques because of their search serendipity—identifying new and perhaps unexpected solutions. A DSS can use such problem solving methods [3] to recommend alternative solutions to human users [13].

The interaction between the DSS and a human user is often initiated by the user who requests a response from the DSS. The DSS could provide more decision guidance if it could also initiate a dialogue to improve the quality of interaction [2]. Utilizing a meta-model which guides the proper use of the decision models provided by a DSS, the user is not required to possess structural knowledge about the problem domain, paving the way for designing next generation DSS [45]. The next generation DSS provides system induced decision guidance (known as Level three DSS) in

* Corresponding author. Tel.: +1-540-231-5869; fax: +1-540-231-2511.
E-mail address: Reza@vt.edu (R. Barkhi).

addition to simple modeling support (Level two DSS) and communication support (Level one DSS) [10]. A Level three DSS [10] should improve the interaction between the DSS and the user and automate the matching of model to problem task. This will free the user from having to understand all the technical details without facing system restrictiveness [6].

The search techniques that have been developed in the past few years, such as genetic algorithms [4,5,14,20,24,26,34,38,42], and tabu search [11,15,19,27,30,36] have been shown to perform very well on many classes of combinatorial optimization problems. These techniques can generate insights that may be difficult to identify using traditional optimization techniques [26]. This is especially true when the complexity of the problems limits the use of traditional optimization techniques [4,5,14] or when finding a satisfactory (or satisficing) solution in a timely fashion is more important than finding an optimal solution.

Although it is valuable to understand that some of the newly developed intelligent search techniques can perform better than traditional solution methods, it is perhaps more insightful to understand the type of problems for which these techniques work well. For example, GA works well in scheduling grouped jobs on a single machine to minimize flow time when the number of fundamental runs is less than the number of jobs to be scheduled [43]. Discovering problem characteristics that suggest solution methods is useful for a DSS that guides the user in utilizing the appropriate modeling technology that fits the task. This can help the designers of next generation DSS to develop decision technologies that provide the best task-technology fit for improved decision guidance [46].

Artificial intelligent search techniques such as GA may improve the generation of diverse alternative solutions [13], serendipitous results, and decision guidance. Despite the usefulness of GA search techniques for problems with certain characteristics, few DSS decision aids can guide the users as to, for example, when to use traditional search techniques and when to use the GA family of search techniques. The GA technique can deal with problems that involve nonlinearity, discontinuity, and uncertainty that make traditional search techniques inappropriate [13]. Utilizing this type of meta-model information, a DSS tool may guide the user to utilize a search technique that is most promising [1]. A Level three DSS should include a meta-model that incorporates the proper use of intelligent search techniques, and execute models that can satisfactorily solve problems that have specific characteristics [38]:

> Exploring why this [subtle relations between problem features and algorithms] should be much more interesting than establishing ranking of algorithms over limited sets of test problems.... And investigating what features of a problem may make it amenable to the recombinative search of genetic algorithms seems like one good place to start...

In the spirit of the preceding statement, this paper studies the effects of various problem characteristics: the tightness of constraints, the linearity of constraints (linear versus nonlinear), and how these properties of the decision problem should be used in a meta-model to choose the appropriate modeling approach. We show how such a meta-model can be used to develop the next generation DSS technology for system induced decision guidance to provide task-technology fit [46].

To illustrate the concepts proposed in this paper, we show how different formulation and solution choices are important in terms of quality of the solution and the efficiency of finding the solution in terms of CPU time. We illustrate these concepts using the *p*-median problem, a problem abstraction that can be used for solving a diverse set of practical problems, including locating emergency vehicles in a city to assure that everyone is covered within a certain time limit, locating warehouses to minimize travel time from manufacturing facilities, locating satellite stations to minimize communication delays and provide a maximum service delay guarantee, locating web servers to improve response time, and placing electronic elements on a printed circuit board to minimize signal traffic on the communication bus. In addition to the practical significance of this problem, the *p*-median problem has several distinct formulations making it possible to compare solutions for various formulations of the same problem. Finally, many other practical and theoretically interesting problems can be formulated as special cases of the *p*-median problem [7–9,16,17,33,35,40,41,44]. Hence, by focusing on this problem, our results are applicable to a large set of problems.

In the next sections we provide a review of the *p*-median problem and its various formulations. After reviewing the basic properties of genetic algorithms in Section 4, we present a simulated environment to inductively capturing meta-models that provide task-technology fit in a DSS model execution library. Two off the shelf Microsoft Excel add-ins are used to demonstrate how one might use known model characteristics to provide the induced decision guidance [46] consistent with a Level three DSS. Finally, we provide conclusions in Section 6.

2. The *p*-median problem example

We use the *p*-median problem as an example for illustrating the concepts proposed in this research. The *p*-median problem is one of the more important problems in the location literature [39]. The objective of the *p*-median problem (PMP) is to identify *p* facility locations in order to minimize the total travel distance that demand must traverse to reach its assigned facility. Given the prevalence of this problem class in both private and public sector location problems, the PMP has received considerable attention in the facility location literature [21,22,28,32].

Hakimi [21,22] proved that if you view the PMP on a network, an optimal solution exists for which all of the facility locations coincide with nodes on the network nodes and not on the arcs. Consequently, for such instances of the PMP, one need only consider a finite number of potential facility sites (i.e., the nodes of the network). This allowed Hakimi [21,22] to formulate the network version of the PMP as a binary integer program to substantially reduce the solution search space. Despite the substantial reduction in the solution search space that results from this innovative solution approach, even the network version of the PMP belongs to the class of problems known as NP-complete [23]. Thus, heuristic algorithms are often used to solve this very complex problem when the size of the problem increases. Among the most common search heuristics are the Add or Drop method [29], and the Exchange method [42]. It is the latter that has been, by far, the most popular [9], since it is robust and computationally efficient. However, this solution method suffers from the problem of any 1-Opt routine in that it may get "trapped" at local optima and therefore does not guarantee a global optimum. A recent area of research that seeks to avoid local optima entrapment in heuristics is tabu search [21]. Rolland et al. [37] developed an efficient tabu search procedure for solving the *p*-median problem.

Another general class of heuristic methods for solving computationally time-consuming problems is Genetic Algorithms (GA). The problem with these heuristics is that the parameters of the heuristics need to be adjusted by trial and error to fit problem characteristics and hence the "art" of fitting the parameters to problem characteristics requires much experience and time-consuming trial and error. While these semi-random search techniques have been shown to efficiently solve some problems once the appropriate fit is developed, it does not uniformly perform better or even as well as traditional methods on many problems.

It would be helpful to identity problem characteristics that affect GA performance and incorporate this meta-model to design a DSS that would guide the users. The DSS will guide the users on which model should be utilized to solve the problem more efficiently, which will lead to higher quality solutions.

3. Formulations of the *p*-median problem

The *p*-median problem can be stated unambiguously as follows: given a graph $G=(V,E)$, we are asked to find a set of nodes, $S=\{s_1,s_2,\ldots,s_p\}$, of size P, where S is a subset of V, such that sum of the distances from the remaining nodes $\{V-S\}$ to the set S is minimized. As a binary integer program, this can be written as:

P-median—linear:

$$\text{Min} \sum_i \sum_j a_i d_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_j x_{ij} = 1 \ \forall \ i \tag{2}$$

$$x_{ij} \leq y_j \forall \ i,j \tag{3}$$

$$\sum_j y_j = P \tag{4}$$

$$x_{ij}, y_j \in \{0,1\} \ \forall \ i,j \tag{5}$$

where

a_i = demand at node i,

d_{ij} = distance from node i to node j,

P = preset number of facilities to locate,

$$x_{ij} = \begin{cases} 1 \text{ if } \textit{node } i \text{ is assigned to facility } j \\ 0 \text{ otherwise} \end{cases},$$

$$y_j = \begin{cases} 1 \text{ if facility } j \text{ is opened} \\ 0 \text{ otherwise} \end{cases}$$

The objective function (1) is linear: it minimizes the weighted sum of distances associated with assigning demand nodes to facility sites. Constraint set (2) ensures that all demand nodes are assigned to exactly one facility. Constraint set (3) prohibits assignment to a facility if it is not opened. The total number of open facility sites is set to P in constraint (4) and the binary nature of the facility siting decision is enforced by constraint set (5). The binary restriction on the x_{ij} variables can easily be relaxed by redefining these variables as the fraction of demand at node i that assigns to a facility at node j. Further, no upper bound constraint on these variables is necessary since constraint (3) automatically enforces a limit of one.

The formulation presented above is referred to as the strong linear programming formulation. Revelle and Swain [35] observed that the strong formulation frequently leads to solutions that are integers. However, when the problem is large in size, it is not efficient to solve the above linear program formulation by the simplex method. This has resulted in the development of special purpose algorithms such as parametric linear programming and special implementation of the simplex method [31], decomposition methods [17], primal gradient algorithm [8], or other heuristic methods such as Lagrangian relaxation [18,28].

We mentioned that one reason for selecting the p-median problem for this study is that we can represent the exact same problem using very tight constraints (i.e., strong formulation) or loose constraints (i.e., weak formulation) while preserving the essential characteristics of the PMP so that the only difference is the tightness of the constraints. An alternative method to formulate this problem is to replace the tight constraints in Eq. (3) with substantially less tight constraints in Eq. (3′) below.

$$\sum_i x_{ij} \leq M y_j \;\; \forall j \tag{3′}$$

In the above inequality M is a very large number as is often used when linking binary and integer variables. This will result in the weak linear programming formulation. This linear program can be solved analytically so that the bound at each node of the enumeration tree could be computed very quickly in constant time [12]. This algorithm has been improved in later studies [25,40]. However, the bounds obtained from the weak linear programming relaxation are generally not sufficiently tight to curtail enumeration adequately.

The weak formulation makes a problem less constrained than does the strong formulation. Hence, when constraint (3) is satisfied so is constraint (3′), but the converse is not true because the solution space of the strong formulation is subsumed by the solution space of the weak formulation. The formulation of the p-median problem that uses constraint (3) has $(j \times i)$ constraints while when we replace Eq. (3) with Eq. (3′) we only have (j) constraints. We will study the effect of the level of constraints in problems formulated as weak versus strong formulations on the performance of traditional LP, and GA. Given that the optimal solution to the PMP is the same regardless of whether the problem is formulated as weak or strong, the PMP provides a reasonable test problem to study the effect of the level of constraints in the problem and the impact of constraints on the solution.

The solution of the PMP can be viewed as consisting of two sets of decisions:

(1) Which P facilities should be open?
(2) Which demand nodes should be assigned to which facility nodes?

For the p-median problem, the decision variables x_{ij} can be trivially found if only the P facilities are known ($y_j = 1$ indicates there is an open facility in location j). When the location of the P facilities is known, each node that is not a facility is connected to its closest facility. Utilizing this insight can substantially reduce the number of decision variables in the problem. There are at most $|V|^2$ decision variables of type x_{ij} and at most $|V|$ of type y_j. Thus, eliminating x_{ij} would reduce the search space for this problem dramatically. For example, for an n node problem, the number of variables would be reduced from n^2 to n. We now introduce a re-formulation of the p-median problem that reduces the number of variables in the problem by using this insight. However, this new formulation introduces a nonlinear constraint that provides us the test problem for studying nonlinearity and the effect it has on the DSS decision guidance. We expect that the nonlinear constraint should not negatively impact the applicability of using a GA as a problem solving method. The GA, and many other meta-heuristic solution techniques, unlike LP solution methods, does not typically require linearity of either the objective function or the constraints. We will also compare the gap in the solution quality and the time to find a solution as a function of linearity of constraints in the traditional NLP and GA methods.

P-median—nonlinear:

$$\text{Min} \sum_{i \in I} a_i c_i \tag{6}$$

s.t.

$$c_i = \underset{j}{\text{Min}}[y_j d_{ij}] \ \forall i \in I \tag{7}$$

$$\sum_{j \in J} y_j = P \tag{8}$$

$$y_j \in \{0, 1\} \tag{9}$$

I is the set of all demand nodes,

$a_i =$ demand at node i,

$d_{ij} =$ distance from node i to node j,

$P =$ preset number of facilities to locate,

$c_i =$ distance from demand node i to closest open facility,

$$y_j = \begin{cases} 1 \text{ if facility } j \text{ is opened} \\ 0 \text{ otherwise} \end{cases}.$$

Note that the objective function now uses constraint set (7) in P-median nonlinear formulation above to determine the distance to the closest open facility node. Given that constraint (7) is nonlinear, this makes the problem a nonlinear programming problem. However, from the perspective of heuristics, such as genetic algorithms, the form of the objective function typically should not substantially impact the solution method as long as the constraints and objective function can be computed. Moreover, the search for the minimum c_i can be performed in linear time, and by introducing one additional cost vector, the total effect of a change in the solution can be performed in $2|V|$ steps. We will see how the GA solves the above nonlinear formulation and the linear version of the same problem to investigate the effect of nonlinearity on GA performance. We provide a brief description of GA in the next section.

4. Theoretical background on genetic algorithms

Genetic Algorithms (GA) are evolutionary search algorithms guided by the principles of evolution and natural selection. The GA searches the solution space as it transforms one solution to another using three operations: Selection, Crossover, and Mutation [20]. The selection operator is sometimes referred to as reproduction.

Each solution is represented as a string—typically a binary string—and is sometimes referred to as a chromosome. The GA operations attempt to generate improved chromosomes in a manner consistent with the concepts of biological evolution and survival of the fittest. The selection operation of a GA generates a set of chromosomes for crossover and mutation. The mutation operation generates new chromosomes by randomly altering the bits (genes) in each chromosome by mutating the bits of one or more existing chromosomes. The crossover operation generates new chromosomes by stochastically exchanging parts of the chromosomes with other strings. Chromosomes are evaluated based on a predefined fitness function that can be viewed as a type of utility or objective function measuring the value or “fitness” of a particular chromosome. The basic idea is that the fitter chromosomes are more likely to be selected and survive into subsequent generations.

The Selection or reproduction selects chromosomes for the next generation. Some common selection strategies are roulette-wheel, rank, tournament and elitist strategies [20]. Rank selection operates by ordering candidate chromosomes according to fitness. Chromosomes are inserted into the next generation as a function of their relative rank. On the other hand, using the roulette-wheel selection, a new chromosome is generated from the crossover of two existing chromosomes as will be explained next. Assume a roulette wheel with k genes each having an equally likely chance of being selected. If the selected gene is between 1 and $c<k$ (which occurs with probability of c/k), then one gene is used; otherwise, with a $(k-c)/k$ chance the other gene is used to generate the new chromosome. This random operation may eliminate "fit" chromosomes. Elitist strategies are used to preserve "fit" chromosomes for future generations. Elitist selection would generate n chromosomes for the next population using roulette-wheel selection. The difference is that if the fittest chromosomes from the previous $(t-1)$ generation are not randomly selected, they are inserted into the new generation (t) manually.

Crossover is the mating operation in the genetic algorithm. Pairs of chromosomes are selected according to the pre-determined crossover rate, χ. Single-point crossover uniformly selects a crossover site on a chromosome. Bits are then exchanged between the chromosome pairs that are positioned to one side of the crossover site. In uniform crossover each gene pair or sub-chromosome is exchanged with probability χ. Crossover can result in an exchange only if the chromosomes being crossed are different.

The Mutation operation ensures that the GA explores new options. New or previously discarded chromosomes are re-introduced into the population through mutation that acts to direct the search into different regions of the search space. Thus, mutation serves to direct the search away from local optima. The most common mutation implementation is uniform mutation. Mutation is performed at the bit (gene) or sub-string level of the chromosomes, rather than at the chromosome or pairs of chromosomes level. Uniform mutation alters each bit or sub-string with pre-determined probability $\mu \in [0.001,0.1]$. Often, the preferred μ setting is found through repeated experimentation.

The GA processes of selection (reproduction), crossover, and mutation continue until some stopping criteria is met. One possible stopping criteria is that a certain number of generations have been produced (for example, 200 generations). As the computational requirements for this type of genetic processing are relatively low given the powerful processors of today, a stopping rule of 50–100 generations is often quite feasible. Obviously, this number is subject to experimental validation, and might also be a function of the particular utility function employed and other problem features.

5. Experimental design and computational results

We used a 2×2 design to investigate how formulation choice influences solution approach; two solution approaches (GA versus LP) and two formulation choices (constraint tightness and objective function nonlinearity). Our focused design will allow us to compare the fit between problem characteristics and solution approaches. Although there are many different solution approaches, e.g. tabu search and neural networks, we focus on only two solution approaches because our objective is to provide a demonstration of how to construct a meta-model, not compare all possible solution approaches.

We predict that the GA method will perform better when the constraints are loose because the solutions that are generated using GA operators are more likely

Table 1
Solutions for the strong formulation

P	LP solution	LP time	GA solution	GA time
	Optimal	CPU	Optimal	CPU
1	15,313	3.0	N/A	N/A
2	6670	5.1	N/A	N/A
3	3624	6.7	N/A	N/A
4	2249	2.5	N/A	N/A
5	1505	2.5	N/A	N/A
6	917	2.4	N/A	N/A
7	452	2.5	N/A	N/A
8	221	2.6	N/A	N/A
9	74	2.3	N/A	N/A
10	35	2.3	N/A	N/A
11	16	2.1	N/A	N/A
12	8	2.0	N/A	N/A
13	0	2.2	N/A	N/A
Average		2.9		

(N/A): The GA did not converge when started with an infeasible solution.

Table 2
Solutions for the weak formulation

P	LP solution	LP time	GA solution	GA time
	Optimal	CPU	Optimal	CPU
1	15,313	2.5	N/A	N/A
2	6670	8.1	N/A	N/A
3	3624	18.0	N/A	N/A
4	2249	26.2	N/A	N/A
5	1505	33.1	N/A	N/A
6	917	28.7	N/A	N/A
7	452	9.9	N/A	N/A
8	221	10.5	N/A	N/A
9	74	6.5	N/A	N/A
10	35	4.4	N/A	N/A
11	16	4.6	N/A	N/A
12	8	4.7	N/A	N/A
13	0	2.5	N/A	N/A
Average		12.3		

(N/A): GA did not converge when started with an infeasible solution.

to be in the feasible set when the constraints are loose (weak formulation) than when they are tight (strong formulation). That is, for example, when a problem is highly constrained, the mutation and crossover operations result in high percentage of solutions that fall in the infeasible region. Also, we predict that GA method will perform better than the traditional methods when the objective function is nonlinear given that GA performance will not degrade when the fitness function is nonlinear. That is, the traditional techniques degrade quickly as we introduce nonlinearity while nonlinearity does not affect GA performance.

To incorporate a meta-model for a DSS to provide system induced decision guidance to users, the DSS runs computational experiments to add to the meta-model of past usage patterns during idle times. This paper shows how two such patterns of problem characteristics, namely the level of constraints in the problem and the linearity of the constraints affect GA performance. We assume the PMP is a test problem and operationalize the level of constraints in the problem as strong versus weak formulation and linearity of the constraints as linear versus nonlinear formulations of the PMP. The DSS uses computational models for each of these problem formulations to generate a new meta-model of the impact of constraint tightness and constraint linearity on GA performance. The DSS solves a large number of problem instances for each formulation and report the average solution gap (i.e., deviation from the optimal solution) and the CPU time over all instances. Because we are interested in studying the impact of these variables on standard LP and GA solution techniques, we use widely available GA and LP Microsoft Excel add-ins: the *What's Best solver* is used for solving LP formulations and Evolver is used to solve the nonlinear problem formulation. All problems are run under Excel (97) under Windows (98) on a 450 MHz Pentium II computer.

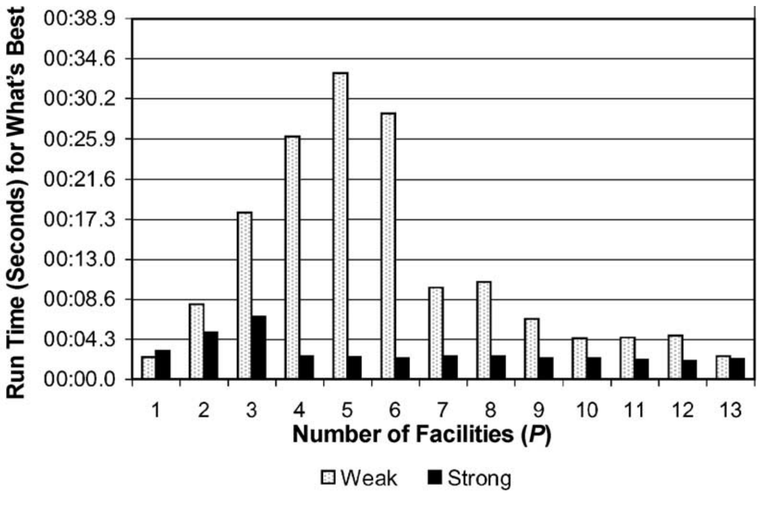

Fig. 1. CPU times for the weak and strong LP formulations.

We experimented with different problem sizes. The computational time for some problems was too large to provide any useful comparisons; for example, some problems never converged or the computational time was several weeks making those size problems not reasonable for comparison purposes for the scope of this study. Hence, we selected a small but well-known test problem consisting of 13 nodes for evaluating the performance of our formulations and solution efforts as has been done in the prior literature [37]. For this problem we varied P from 1 to 13, in effect creating 13 problem instances for each of the strong and weak formulations. For the LP formulations we ran the *What's Best solver* once for each formulation problem instance (26 runs in total).

The solution values and CPU times corresponding to the strong formulation are given in Table 1. The corresponding information resulting from the weak formulation is summarized in Table 2. In these tables, column 1 displays the number of open facilities. Column 2 lists the optimal solution found by the *What's Best solver*. Column 3 is the CPU times. The last columns with values that are indicated as not applicable indicate that the problems were not solved using the indicated technique because that method failed to solve the problem in a reasonable time for comparison purposes. In other words, the GA could not solve the problems formulated with linear objective function and constraints shown in strong and weak formulation while the traditional search techniques find the solutions in reasonable time.

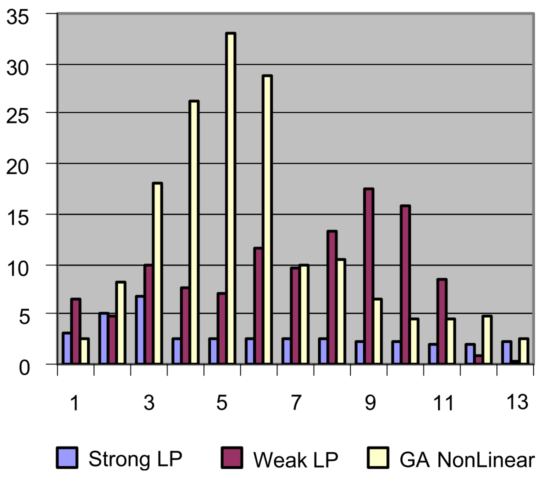

Fig. 2. Comparison of CPU times for the LP and the GA formulations for p-median problem.

The CPU times from the LP solutions (Tables 1 and 2) are summarized in Fig. 1. It shows that the LP weak formulation requires on average almost an order of magnitude more computational effort than the LP strong formulation (for problems were $1 < P < 13$). For the strong formulation, the CPU requirements are the highest when $P = 2$ and $P = 3$. For the weak formulation, this happens when $3 \leq P \leq 6$. The variance in computational time of solution methods is substan-

Table 3
Solutions for the nonlinear formulation

P		GA best	GA average	GA average	GA S.D.	GA average time	GA time	LP best	LP time
	Optimal	Solution	Solution	Gap (%)	Solution	CPU (min)	S.D. time to best	Solution	CPU
1	15,313	15,313	16,263.4	6.2	1530.3	6.4	6.1	N/A	N/A
2	6670	6670	7474.3	12.1	1339.4	4.9	4.1	N/A	N/A
3	3624	3624	3624.0	0.0	0.0	9.8	6.0	N/A	N/A
4	2249	2249	2536.0	12.8	303.8	7.5	7.4	N/A	N/A
5	1505	1505	1900.5	26.3	335.8	7.0	5.5	N/A	N/A
6	917	917	1002.0	9.3	138.0	11.7	5.3	N/A	N/A
7	452	452	516.5	14.3	103.9	9.7	6.0	N/A	N/A
8	221	221	236.3	6.9	24.6	13.4	11.8	N/A	N/A
9	74	74	86.8	17.3	40.5	17.5	5.8	N/A	N/A
10	35	35	35.0	0.0	0.0	15.9	9.9	N/A	N/A
11	16	16	16.0	0.0	0.0	8.4	5.3	N/A	N/A
12	8	8	8.0	0.0	0.0	0.9	3.7	N/A	N/A
13	0	0	0.0	0.0	0.0	0.2	0.6	N/A	N/A
Average				8.1		8.7			

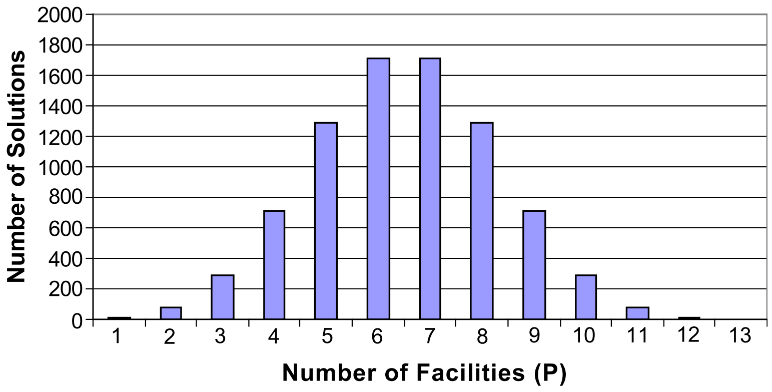

Fig. 3. Solution space for nonlinear problems.

tially higher with the weak formulation than it is with the strong formulation. These results illustrate that for traditional LP solution methodologies, the choice of formulation significantly influences CPU requirements and solution times and variances. In addition, some formulation choices make it impossible to solve the problem using meta-heuristics such as the GA technique in a reasonable time. This is an example of the kind of information that a meta-model might store and use for model selection and system induced decision guidance.

Table 3 summarizes the results from applying a simple GA to the nonlinear formulation of the PMP. The results in this table are based on solving each problem instance 10 times using Evolver. Given that GA is a heuristic search and may not result in the optimal solution and that each run may be subject to randomness, we report the average solution and CPU time for the 10 problem instances. Table 3 presents the best (column 3) and average (column 4) solutions as well as the average solution time (column 7) to find the best solution for each problem instance. The number of facilities and known optimal solutions are reported in columns 1 and 2, respectively, and standard deviations for the CPU times are reported in column 8. Columns 8 and 9 show that this nonlinear problem was not solved using the traditional simplex method because of the nonlinearity of the objective function. While the simplex method cannot solve this nonlinear problem, the GA effectively deals with the nonlinearity of the objective function and can easily solve the problem in a reasonable time. This information could also be incorporated into the meta-model repository to provide system induced decision guidance for similar problems in the future.

While utilizing GA, we used the default Evolver parameters to maintain the focus on the impact of constraint tightness as well as the linearity of constraints on GA performance. The population size was kept at 50, the crossover rate was 0.5, and the mutation rate was 0.06.[1] Evolver was programmed to end if no improvement in solution quality was found in 10,000 iterations. The GA started with a random feasible initial solution.

Table 3 illustrates that the GA searching the solution space of the nonlinear formulation of the problem always found the optimal solution at least once for each problem instance, resulting in an average optimality gap of 0% (Best Solution, column 3). The average optimality gap over all obtained solutions (i.e., not only the best solutions for each instance) is 8.1%.

Fig. 2 compares the CPU requirements for solving the GA procedure based on the nonlinear formulation and with the same requirements for the weak and strong linear formulations. The results suggest that formulating the problem as a nonlinear problem that can utilize the capabilities of a GA to deal with nonlinearity can solve the problem faster than traditional methods even when the GA param-

[1] It is outside the scope of this paper to attempt to experiment with and tailor these parameters given that the focus of the study is to investigate constraint tightness and linearity on GA performance.

eters are not “customized for the specific problem domain.”

To gain an understanding about the relationship between traditional search and solution space size and GA technique and solution space size, we calculate the number of potential solutions in the solution space of the linear and the nonlinear problems of a 13-node PMP. The number of possible solutions to each problem instance when problem is formulated as a nonlinear problem is depicted in Fig. 3. The number of possible solutions for the nonlinear formulation is $\binom{13}{P}$, since we only need to select P open facilities out of the 13 total possible facility sites. For the linear formulation, the corresponding solution space is determined by the formula $\binom{13}{P}*(V-P)*\binom{P}{1}$, since the facility assignments also have to be determined independently from the facility opening decision. This solution space for the linear formulation is depicted in Fig. 4.

Figs. 3 and 4 illustrate that while the pattern of the solution space is identical for the linear and the nonlinear formulations (peaks at $P=6$ and $P=7$), the size of the solution space for the linear formulation is much greater than it is for the nonlinear formulation. Despite the similarity in the solution space patterns, the CPU requirements of the LP weak, LP strong and the GA are not similar. For example, referring back to Fig. 2, the peak CPU requirements for the GA occurs at $P=9$ and $P=10$. This is not consistent with the peak in solution space size that occurs at $P=6$ and $P=7$ shown in Fig. 3. Hence, when applying GA to the PMP, the search is not only influenced by the size of the solution space, but also by the increased number of infeasible solutions. This was clearly illustrated when we applied the GA to the linear formulations (both strong and weak). For these problem formulations, even though the GA was provided with a feasible starting solution, it did never yield a better than initial solution even after 50 million iterations (several days of computing time) and hence the solution columns received a value of N/A. The tight relationship between the decision variables (i.e., linear formulations) can be a source of increased infeasibility, which leads to the conclusion that a GA is not an appropriate methodology for such formulations when everything else is held constant. Thus, both constraint tightness and constraint linearity seems to substantially influence GA performance. The kind of experiments described above generates the meta-model that will be captured in a repository inductively and is used deductively for future interaction with DSS that provides system induced decision guidance.

6. Conclusions

In this paper, we have used an example to illustrate how the meta-model stored in a repository of a DSS can provide system induced decision guidance for the users. This type of meta-model is generated from experiments that study how problem characteristics can suggest certain solution approach over others. The meta-model is generated inductively and once stored can be used deductively when similar situations arise in the future. We showed how a meta-model is generated by focusing on a specific problem and

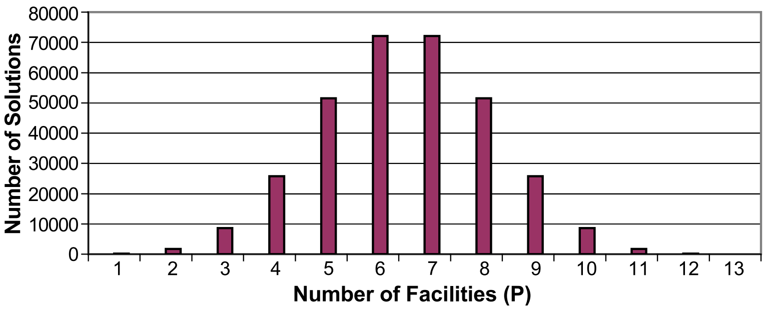

Fig. 4. Solution space for linear problems.

investigating the impact of problem characteristics: constraint tightness, and constraint linearity, on the performance of various optimization and GA solution techniques. An example of the experiments that generate the meta-model information in this paper includes the following:

1. Constraint tightness and constraint linearity can have substantial influence on CPU requirements for the *p*-median problem.
2. For the linear programming solution approach, the strong formulation of the PMP leads to lower CPU times.
3. PMP could be effectively solved by a genetic algorithm approach if formulated as a nonlinear problem.
4. The PMP that is formulated as a nonlinear problem cannot be solved easily using traditional optimization techniques.

In this paper, we generated the meta-model which shows that using the nonlinear formulation of the problem yields several advantages that will substantially improve the GA performance:

1. The number of decision variables is reduced from n^2+n to n,
2. Provides a representation that results in less infeasible solutions, hence enabling the GA to achieve better solutions with acceptable CPU times.

The solution of GA technique can be more efficient if the problem is formulated to reduce the tightness of the constraints so that more feasible chromosomes are generated as a result of selection, crossover, and mutation. In addition, GA performance can be more efficient if the number of decision variables is smaller and the interaction among decision variables is less constrained. The nonlinearity of the constraints that converts simple LP problems into complex NLP problems is not an issue when the problems are solved using GA technique. Hence, a specific meta-model to guide DSS users would be similar to the following insight:

> GA performance improves if you model a nonlinear formulation that reduces constraint tightness and the number of interactions among decision variables.

This type of information that is generated from experiments become the meta-model that provides system induced decision guidance for DSS users. This meta-model is derived inductively as rules that will be applied to specific problems deductively. DSS usage encounters different types of problem characteristics and results in a rich meta-model base of problem characteristics that lend themselves to evolutionary solution techniques. This way the DSS can guide users on how to solve problems efficiently and effectively by providing task-technology fit.

A DSS that provides system induced decision guidance does not require a user with structural knowledge about how best to formulate and solve problems. Hence, a manager who wants to solve a complex problem by interacting with a Level three DSS need not necessarily be an expert analyst equipped with knowledge of decision technology and its intricacies. Although we only compared two different solution techniques and two different problem characteristics, the two-by-two experimental study shows how the results of experimentation can be used as a meta-model to provide system induced decision guidance.

References

[1] R. Barkhi, The effects of decision guidance and problem modeling on group decision-making, J. Manage. Inf. Syst. 18 (3) (2002) 259–282.

[2] M. Beynon, S. Rasmequan, S. Russ, A new paradigm for computer-based decision support, Decis. Support Syst. 33 (2002) 127–142.

[3] N. Bolloju, M. Khalifa, E. Turban, Integrating knowledge management into enterprise environments for the next generation decision support, Decis. Support Syst. 22 (2002) 163–176.

[4] R. Cheng, G. Mitsuo, Y. Tsujimura, A tutorial survey of job-shop scheduling problems using genetic algorithms: Part II. Hybrid genetic search strategies, Comput. Ind. Eng. 36 (1999) 343–364.

[5] N.C. Chiu, S.C. Fang, Y.S. Lee, Seqencing parallel machining operations by genetic algorithms, Comput. Ind. Eng. 36 (1999) 259–280.

[6] P.C. Chu, J. Elam, Induced system restrictiveness: an experi-

mental demonstration, IEEE Trans. Syst. Man Cybern. SMC-20 (1) (1990) 195–201.

[7] G. Cornuejols, J.M. Thizy, Some facets of the simple plant location polytope, SIAM J. Algebr. Discrete Methods 3 (1982) 504–510.

[8] G. Cornuejols, M.L. Fisher, G.L. Nemhauser, Location of bank accounts to optimize float: an analytical study of exact and approximate algorithms, Manage. Sci. 23 (1977) 789–810.

[9] P.J. Densham, G. Rushton, A more efficient heuristic for solving large *p*-median problems, papers in regional science, J. RSAI 71 (3) (1992) 307–329.

[10] G. DeSanctis, R.B. Gallupe, A foundation for the study of group decision support systems, Manage. Sci. 33 (5) (1987) 589–609.

[11] D. DeWerra, A. Hertz, Tabu search techniques: a tutorial and an application to neural networks, OR-Spektrum 11 (1989) 131–141.

[12] M.A. Efroymson, T.L. Ray, A branch and bound algorithm for plant location, Oper. Res. 14 (1996) 361–368.

[13] B. Fazlollahi, R. Vahidov, A method for generation of alternatives by decision support systems, J. Manage Inf. Syst. 18 (2) (2001) 229–250.

[14] S.A. Feysbaksh, M. Matsui, Adam–Eve-like genetic algorithm: a methodology for optimal design of a simple flexible assembly system, Comput. Ind. Eng. 36 (1999) 233–258.

[15] C. Friden, A. Hertz, D. deWerra, Stabulus: a technique for finding stable sets in large graphs with tabu search, Computing 42 (1989) 35–44.

[16] R.D. Galvão, A dual-bounded algorithm for the *p*-median problem, Oper. Res. 28 (1980) 1112–1121.

[17] R.S. Garfinkel, A.W. Neebe, M.R. Rao, An algorithm for the m-median plant location problem, Transport. Sci. 8 (1974) 217–236.

[18] A.M. Geoffrion, Lograngian relaxation for integer programming, Math. Program. Stud. 2 (1974) 82–114.

[19] F. Glover, M. Laguna, Tabu search, in: C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, Blackwell, 1993.

[20] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[21] S.L. Hakimi, Optimal locations of switching centers and the absolute centers and the medians of a graph, Oper. Res. 12 (1964) 450–459.

[22] S.L. Hakimi, Optimum distribution of switching centers in a communications network and some related graph theoretic problems, Oper. Res. 13 (1965) 462–475.

[23] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems: Part 2. The *p*-medians, SIAM J. Appl. Math. 37 (1979) 539–560.

[24] A. Kershenbaum, When genetic algorithms work best, INFORMS J. Comput. 9 (3) (1997) 254–255.

[25] B.M. Khumawala, An efficient branch and bound algorithm for the warehouse location problem, Manage. Sci. 18 (1972) 718–731.

[26] B.R. Klemz, Using genetic algorithms to assess the impact of pricing activity timing, Omega Int. J. Manage. Sci. 27 (3) (1999) 363–372.

[27] J.G. Klincewicz, Avoiding local optima in the p-hub location problem using tabu search and grasp, Ann. Oper. Res. 40 (1992) 121–132.

[28] J. Krarup, P.M. Pruzan, The simple plant location problem: survey and synthesis, Eur. J. Oper. Res. 12 (1983) 36–81.

[29] A.A. Kuehn, M.J. Hamburger, A heuristic program for locating warehouses, Manage. Sci. 9 (1963) 643–666.

[30] M. Laguna, J.W. Barnes, F. Glover, Tabu search for a single machine scheduling problem, J. Intell. Manuf. 2 (1991) 253–260.

[31] R.E. Marsten, An Algorithm for Finding Almost All the Medians of a Network. Discuss. Pap. 23. Center for Mathematical Studies in Econometrics and Management Science, Northwestern University, Evanston, IL, 1972.

[32] P.B. Mirchandani, J.M. Reilly, Spatial nodes in discrete location problems, in: A. Gosh, G. Rushton (Eds.), Spatial Analysis and Location-Allocation Models, Van Nostrand-Reinhold, New York, 1987, pp. 186–223.

[33] S.C. Narula, U.I. Ogbu, H.M. Samuelson, An algorithm for the *p*-median problem, Oper. Res. 25 (1977) 709–713.

[34] C.R. Reeves, Genetic algorithms of the operations researcher, INFORMS J. Comput. 9 (3) (1997) 231–250.

[35] C.S. Revelle, R.S. Swain, Central facilities location, Geogr. Anal. 2 (1970) 30–42.

[36] E. Rolland, H. Pirkul, F. Glover, Tabu search for graph partitioning, Ann. Oper. Res. 63 (1996) 209–232.

[37] E. Rolland, D.A. Schilling, J.R. Current, An efficient tabu search procedure for the *p*-median problem, Eur. J. Oper. Res. 96 (1997) 329–342.

[38] P. Ross, What are genetic algorithms good at? INFORMS J. Comput. 9 (3) (1997) 260–262.

[39] D.A. Schilling, V. Jayaraman, R. Barkhi, A review of covering problems in facility location, Location Sci. 1 (1) (1993) 25–55.

[40] K. Spielberg, Plant location with generalized search origin, Manage. Sci. 16 (1969) 165–178.

[41] B.C. Tansel, R.L. Francis, T.J. Lowe, Location on networks: a survey: Part I. The *p*-center and *p*-median problems, Manage. Sci. 29 (4) (1983).

[42] M.B. Teitz, P. Bart, Heuristic methods for estimating the generalized vertex median of a weighted graph, Oper. Res. 16 (1968) 955–961.

[43] D. Wang, M. Gen, R. Cheng, Scheduling grouped jobs on single machine with genetic algorithm, Comput. Ind. Eng. 36 (1999) 309–324.

[44] J.R. Weaver, R. Church, A median location model with nonclosest facility services, Transport. Sci. 19 (1) (1985) 58–74.

[45] L.A. West, T.J. Hess, Metadata as a knowledge management tool: supporting intelligent agent and end user access to spatial data, Decis. Support Syst. 32 (2002) 247–264.

[46] I. Zigurs, B.K. Buckland, A theory of task/technology fit and group support systems effectiveness, Q. - Mus. Inf. Serv. 22 (3) (1998) 313–334.

Reza Barkhi is an Associate Professor of Information Systems and an Alumni Research Fellow in the Department of Accounting and Information Systems, Pamplin College of Business, at Virginia Polytechnic Institute and State University. His current research interests are in the areas of collaborative technologies and managerial problem-solving, and Application of Artificial Intelligence Techniques to Organizational problems. Reza has published in journals such as *Journal of Management Information Systems*, *Location Science*, *Computers and Operations Research*, *Group Decision & Negotiation*, *Information Resource Management Journal*, *International Journal of Information Technology and Decision Making*, *Journal of Advanced manufacturing Systems*, *and European Journal of Operational Research*. He is on the Editorial Board of *Computers & OR* Journal. He received a BS in Computer Science, an MBA, MA, and a PhD in Information Systems and Decision Sciences from The Ohio State University.

Erik Rolland is Associate Professor and academic area coordinator for the Information Systems group at the A. Gary Anderson Graduate School of Management, University of California at Riverside. His publications appear in journals such as *Operations Research*, *Transportation Sciences*, the *European Journal of Operational Research*, *Annals of Operations Research*, *Decision Support Systems*, and *Computers and Operations Research*. Erik received his BS in Computer Sciences, MA in Business Administration, and PhD in MIS/Decision Sciences, all from the Ohio State University.

John Butler is an Assistant Professor of Management Information Systems at the Ohio State University. His research, teaching and consulting focus on the use of the use of quantitative models and technology to improve decision-making. His research has appeared in leading journals, including, *Management Science* and *Operations Research* and he is currently on the editorial board of *Decision Support Systems*. He is currently a council member of the INFORMS Decision Analysis Society and also serves as Education Editor for the society's web materials. John received his BBA in Management Information Systems from Texas A&M University and PhD in Management Science and Information Systems from the University of Texas at Austin.

Weiguo Fan is an Assistant Professor of information systems and computer science at the Virginia Polytechnic Institute and State University. He received his PhD in Information Systems from the University of Michigan Business School, Ann Arbor, in July 2002, a MSce in Computer Science from the National University of Singapore in 1997, and a BE in Information and Control Engineering from the Xi'an Jiaotong University, P.R. China, in 1995. His research interests include data mining, text/web mining, web computing, business intelligence, and knowledge management. His research has appeared in many prestigious information technology journals such as *Information Processing and Management*, *IEEE Transactions on Knowledge and Data Engineering*, *Information Systems*, *Decision Support Systems*, *ACM Transactions on Internet Technology*, *Journal of the American Society on Information Science and Technology*, *Journal of Classification, International Journal of Electronic Business*, and in conferences such as ICIS, HICSS, AMCIS, WWW, CIKM, DS, ICOTA.